

# **Exhibit 18**

# MEMORY SYSTEMS

Cache, DRAM, Disk



BRUCE JACOB • SPENCER W. NG • DAVID T. WANG

# Overview of DRAMs

DRAM is the “computer memory” that you order through the mail or purchase at the store. It is what you put more of into your computer as an upgrade to improve the computer’s performance. It appears in most computers in the form shown in Figure 7.1—the ubiquitous *memory module*, a small computer board (a *printed circuit board*, or *PCB*) that has a handful of chips attached to it. The eight black rectangles on the pictured module are the DRAM chips: plastic packages, each of which encloses a *DRAM die* (a very thin, fragile piece of silicon).

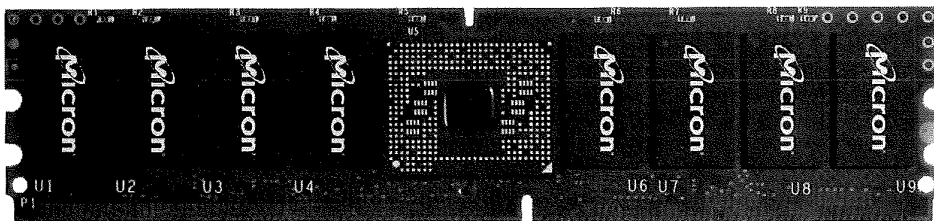
Figure 7.2 illustrates DRAM’s place in a typical PC. An individual DRAM device typically connects indirectly to a CPU (i.e., a microprocessor) through a memory controller. In PC systems, the memory controller is part of the *north-bridge* chipset that handles potentially multiple microprocessors, the graphics co-processor, communication to the *south-bridge* chipset (which, in turn, handles all of the system’s I/O functions), as well as the interface to the DRAM system. Though still often referred to as “chipsets”

these days, the north- and south-bridge chipsets are no longer sets of chips; they are usually implemented as single chips, and in some systems the functions of both are merged into a single die.

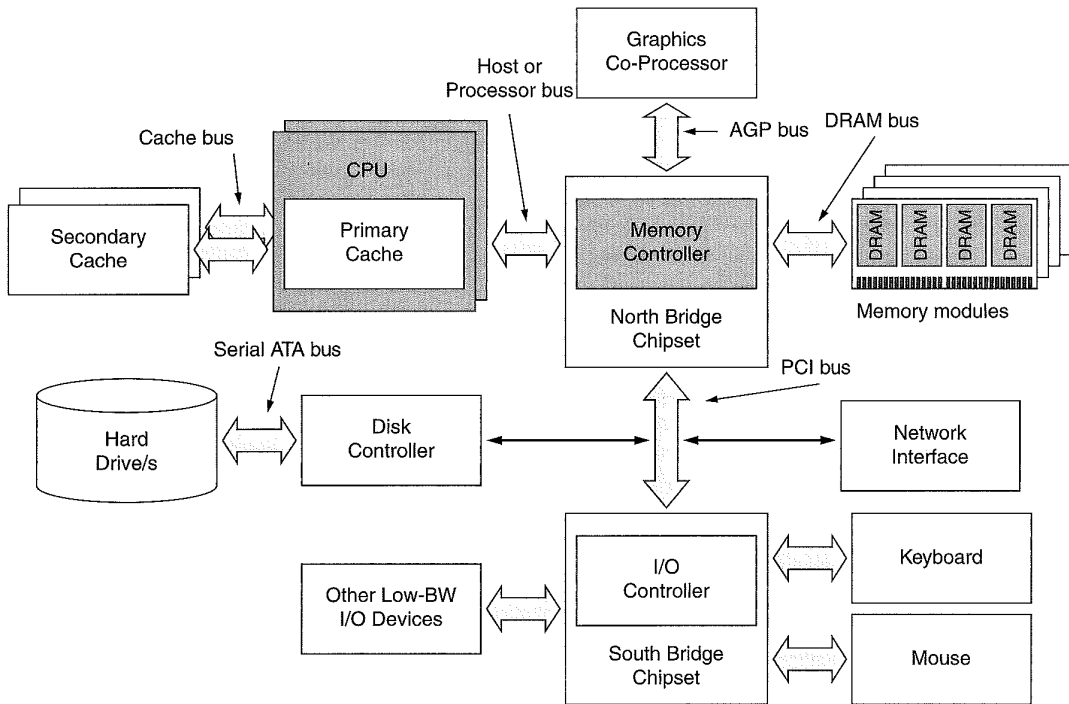
Because DRAM is usually an external device by definition, its use, design, and analysis must consider effects of implementation that are often ignored in the use, design, and analysis of on-chip memories such as SRAM caches and scratch-pads. Issues that a designer must consider include the following:

- Pins (e.g., their capacitance and inductance)
- Signaling
- Signal integrity
- Packaging
- Clocking and synchronization
- Timing conventions

Failure to consider these issues when designing a DRAM system is guaranteed to result in a sub-optimal, and quite probably non-functional, design.



**FIGURE 7.1:** A memory module. A memory module, or DIMM (dual in-line memory module), is a circuit board with a handful of DRAM chips and associated circuitry attached to it.



**FIGURE 7.2:** A typical PC organization. The DRAM subsystem is one part of a relatively complex whole. This figure illustrates a two-way multi-processor, with each processor having its own dedicated secondary cache. The parts most relevant to this report are shaded in darker grey: the CPU, the memory controller, and the individual DRAMs.

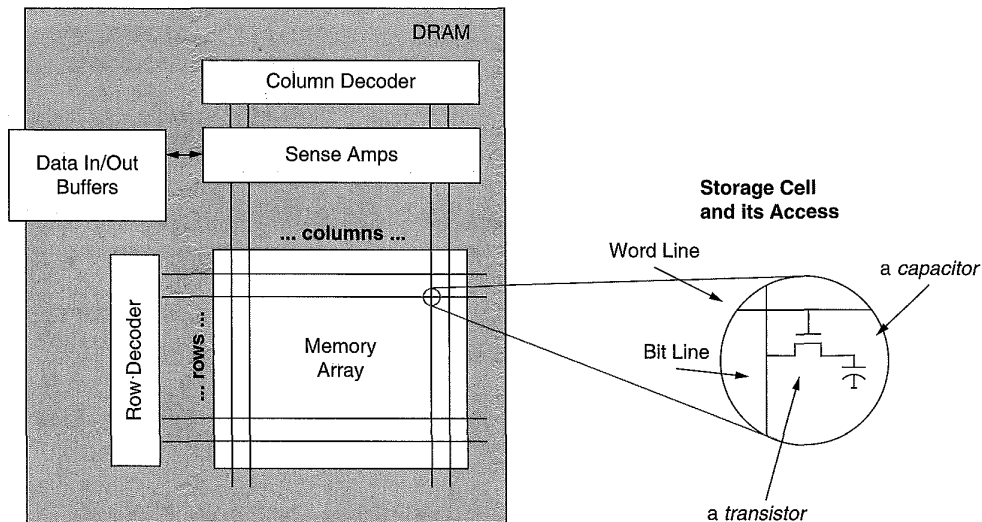
Thus, much of this section of the book deals with low-level implementation issues that were not covered in the previous section on caches.

## 7.1 DRAM Basics: Internals, Operation

A random-access memory (RAM) that uses a single transistor-capacitor pair for each bit is called a dynamic random-access memory or DRAM. Figure 7.3 shows, in the bottom right corner, the circuit for the storage cell in a DRAM. This circuit is *dynamic* because the capacitors storing electrons are not perfect devices, and their eventual leakage requires that, to retain information stored there, each

capacitor in the DRAM must be periodically *refreshed* (i.e., read and rewritten).

Each DRAM die contains one or more *memory arrays*, rectangular grids of storage cells with each cell holding one bit of data. Because the arrays are rectangular grids, it is useful to think of them in terms associated with typical grid-like structures. A good example is a Manhattan-like street layout with avenues running north-south and streets running east-west. When one wants to specify a rendezvous location in such a city, one simply designates the intersection of a street and an avenue, and the location is specified without ambiguity. Memory arrays are organized just like this, except where Manhattan is organized into *streets* and *avenues*, memory arrays are organized



**FIGURE 7.3:** Basic organization of DRAM internals. The DRAM memory array is a grid of storage cells, where one bit of data is stored at each intersection of a *row* and a *column*.

into *rows* and *columns*. A DRAM chip's memory array with the rows and columns indicated is pictured in Figure 7.3. By identifying the intersection of a row and a column (by specifying a *row address* and a *column address* to the DRAM), a memory controller can access an individual storage cell inside a DRAM chip so as to read or write the data held there.

One way to characterize DRAMs is by the number of memory arrays inside them. Memory arrays within a memory chip can work in several different ways. They can act in unison, they can act completely independently, or they can act in a manner that is somewhere in between the other two. If the memory arrays are designed to act in unison, they operate as a unit, and the memory chip typically transmits or receives a number of bits equal to the number of arrays each time the memory controller accesses the DRAM. For example, in a simple organization, a x4 DRAM (pronounced “by four”) indicates that the DRAM has at least four memory arrays and that a column width is 4 bits (each column read or write transmits 4 bits of data). In a x4 DRAM part, four arrays each read 1 data

bit in unison, and the part sends out 4 bits of data each time the memory controller makes a column read request. Likewise, a x8 DRAM indicates that the DRAM has at least eight memory arrays and that a column width is 8 bits. Figure 7.4 illustrates the internal organization of x2, x4, and x8 DRAMs. In the past two decades, wider output DRAMs have appeared, and x16 and x32 parts are now common, used primarily in high-performance applications.

Note that each of the DRAM illustrations in Figure 7.4 represents multiple arrays but a single *bank*. Each set of memory arrays that operates independently of other sets is referred to as a bank, not an array. Each bank is independent in that, with only a few restrictions, it can be activated, precharged, read out, etc. at the same time that other banks (on the same DRAM device or on other DRAM devices) are being activated, precharged, etc. The use of multiple independent banks of memory has been a common practice in computer design since DRAMs were invented. In particular, *interleaving* multiple memory banks has been a popular method used to achieve high-bandwidth memory busses using



**FIGURE 7.4:** Logical organization of wide data-out DRAMs. If the DRAM outputs more than one bit at a time, the internal organization is that of multiple arrays, each of which provides one bit toward the aggregate data output.

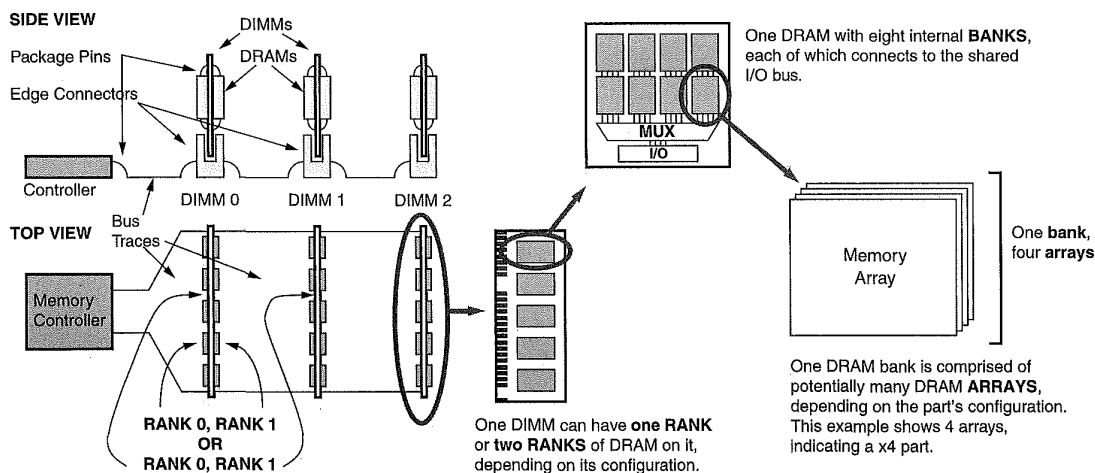
low-bandwidth devices. In an interleaved memory system, the data bus uses a frequency that is faster than any one DRAM bank can support; the control circuitry toggles back and forth between multiple banks to achieve this data rate. For example, if a DRAM bank can produce a new chunk of data every 10 ns, one can toggle back and forth between two banks to produce a new chunk every 5 ns, or round-robin between four banks to produce a new chunk every 2.5 ns, thereby effectively doubling or quadrupling the data rate achievable by any one bank. This technique goes back at least to the mid-1960s, where it was used in two of the highest performance (and, as it turns out, best documented) computers of the day: the IBM System/360 Model 91 [Anderson et al. 1967] and Seymour Cray's Control Data 6600 [Thornton 1970].

Because a system can have multiple DIMMs, each of which can be thought of as an independent bank, and the DRAM devices on each DIMM can implement internally multiple independent banks, the word “rank” was introduced to distinguish DIMM-level independent operation versus internal-bank-level independent operation. Figure 7.5 illustrates the various levels of organization in a modern DRAM system. A system is composed of potentially many independent DIMMs. Each DIMM may contain one

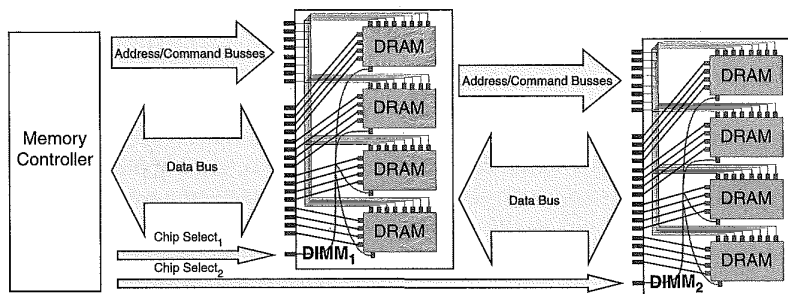
or more independent ranks. Each rank is a set of DRAM devices that operate in unison, and internally each of these DRAM devices implements one or more independent banks. Finally, each bank is composed of slaved memory arrays, where the number of arrays is equal to the data width of the DRAM part (i.e., a x4 part has four slaved arrays per bank). Having concurrency at the rank and bank levels provides bandwidth through the ability to pipeline requests. Having multiple DRAMs acting in unison at the rank level and multiple arrays acting in unison at the bank level provides bandwidth in the form of parallel access.

The busses in a JEDEC-style organization are classified by their function and organization into *data*, *address*, *control*, and *chip-select* busses. An example arrangement is shown in Figure 7.6, which depicts a memory controller connected to two memory modules. The data bus that transmits data to and from the DRAMs is relatively wide. It is often 64 bits wide, and it can be much wider in high-performance systems. A dedicated address bus carries row and column addresses to the DRAMs, and its width grows with the physical storage on a DRAM device (typical widths today are about 15 bits). A control bus is composed of the row and column strobes,<sup>1</sup> output enable, clock, clock enable, and other related signals. These

<sup>1</sup>A “strobe” is a signal that indicates to the recipient that another signal, e.g., data or command, is present and valid.



**FIGURE 7.5:** DIMMs, ranks, banks, and arrays. A system has potentially many DIMMs, each of which may contain one or more ranks. Each rank is a set of ganged DRAM devices, each of which has potentially many banks. Each bank has potentially many constituent arrays, depending on the part's data width.

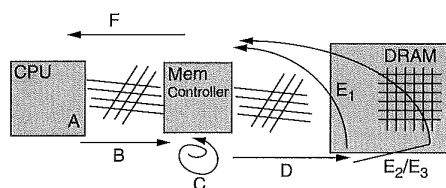


**FIGURE 7.6:** JEDEC-style memory bus organization. The figure shows a system of a memory controller and two memory modules with a 16-bit data bus and an 8-bit address and command bus.

signals are similar to the address-bus signals in that they all connect from the memory controller to every DRAM in the system. Finally, there is a chip-select network that connects from the memory controller to every DRAM in a *rank* (a separately addressable set of DRAMs). For example, a memory module can contain two ranks of DRAM devices; for every DIMM in the system, there can be two separate chip-select

networks, and thus, the size of the chip-select “bus” scales with the maximum amount of physical memory in the system.

This last bus, the chip-select bus, is essential in a JEDEC-style memory system, as it enables the intended recipient of a memory request. A value is asserted on the chip-select bus at the time of a request (e.g., read or write). The chip-select bus



A: Transaction request may be delayed in Queue  
 B: Transaction request sent to Memory Controller  
 C: Transaction converted to Command Sequences  
 (may be queued)

D: Command/s Sent to DRAM

E<sub>1</sub>: Requires only a **CAS** or

E<sub>2</sub>: Requires **RAS + CAS** or

E<sub>3</sub>: Requires **PRE + RAS + CAS**

F: Transaction sent back to CPU

DRAM Latency = A + B + C + D + E + F

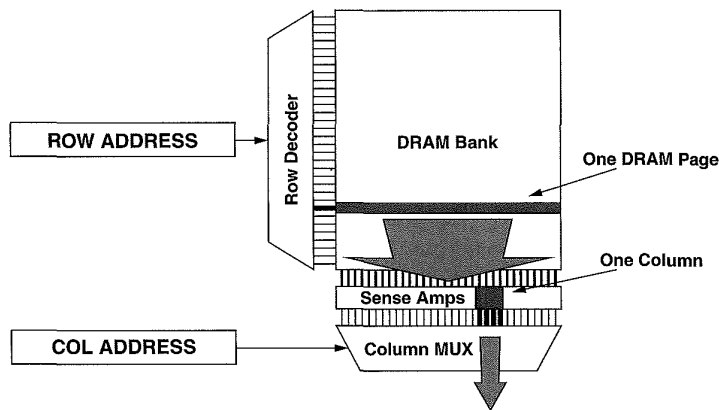
**FIGURE 7.7:** System organization and the steps of a DRAM read. Reading data from a DRAM is not as simple as an SRAM, and at several of the stages the request can be stalled.

contains a separate wire for every rank of DRAM in the system. The chip-select signal passes over a wire unique to each small set of DRAMs and enables or disables the DRAMs in that rank so that they, respectively, either handle the request currently on the bus or ignore the request currently on the bus. Thus, only the DRAMs to which the request is directed handle the request. Even though all DRAMs in the system are connected to the same address and control buses and could, in theory, all respond to the same request at the same time, the chip-select bus prevents this from happening.

Figure 7.7 focuses attention on the microprocessor, memory controller, and DRAM device and illustrates the steps involved in a DRAM request. As mentioned previously, a DRAM device connects indirectly to a microprocessor through a memory controller; the microprocessor connects to the memory controller through some form of network (bus, point-to-point, crossbar, etc.); and the memory controller connects to the DRAM through another network (bus, point-to-point, etc.). The memory controller acts as a liaison between the microprocessor and DRAM so that the microprocessor does not need to know the details of the

DRAM's operation. The microprocessor presents requests to the memory controller that the memory controller satisfies. The microprocessor connects to potentially many memory controllers at once; alternatively, many microprocessors could be connected to the same memory controller. The simplest case (a *uniprocessor* system) is illustrated in the figure. The memory controller connects to potentially many DRAM devices at once. In particular, DIMMs are the most common physical form in which consumers purchase DRAM, and these are small PCBs with a handful of DRAM devices on each. A memory controller usually connects to at least one DIMM and, therefore, multiple DRAM devices at once.

Figure 7.7 also illustrates the steps of a typical DRAM read operation. After ordering and queueing requests, the microprocessor sends a given request to the memory controller. Once the request arrives at the memory controller, it is queued until the DRAM is ready and all previous and/or higher priority requests have been handled. The memory controller's interface to the DRAM is relatively complex (compared to that of an SRAM, for instance); the row-address strobe (RAS) and column-address strobe (CAS) components are shown in detail in Figure 7.8. Recall from Figure 7.3 that the capacitor lies at the intersection of a wordline and a bitline; it is connected to the bitline through a transistor controlled by the wordline. A transistor is, among other things, a switch, and when the voltage on a wordline goes high, all of the transistors attached to that wordline become closed switches (turned on), connecting their respective capacitors to the associated bitlines. The capacitors at each intersection of wordline and bitline are extremely small and hold a number of electrons that are minuscule relative to the physical characteristics of those bitlines. Therefore, special circuits called *sense amplifiers* are used to detect the values stored on the capacitors when those capacitors become connected to their associated bitlines. The sense amplifiers first *precharge* the bitlines to a voltage level that is halfway between logic level 0 and logic level 1. When the capacitors are later connected to the bitlines through the transistors, the capacitors change the voltage levels on those bitlines very slightly. The sense amplifiers detect the minute changes and pull



**FIGURE 7.8:** The multi-phase DRAM-access protocol. The row access drives a DRAM page onto the bitlines to be sensed by the sense amps. The column address drives a subset of the DRAM page onto the bus (e.g., 4 bits).

the bitline voltages all the way to logic level 0 or 1. Bringing the voltage on the bitlines to fully high or fully low, as opposed to the precharged state between high and low, actually recharges the capacitors as long as the transistors remain on.

Returning to the steps in handling the read request. The memory controller must decompose the provided data address into components that identify the appropriate rank within the memory system, the bank within that rank, and the row and column inside the identified bank. The components identifying the row and column are called the *row address* and the *column address*. The bank identifier is typically one or more address bits. The rank number ends up causing a chip-select signal to be sent out over a single one of the separate chip-select lines.

Once the rank, bank, and row are identified, the bitlines in the appropriate bank must be *precharged* (set to a logic level halfway between 0 and 1). Once the appropriate bank has been precharged, the second step is to *activate* the appropriate row inside the identified rank and bank by setting the chip-select signal to activate the set of DRAMs comprising the

desired bank, sending the row address and bank identifier over the address bus, and signaling the DRAM's  $\overline{\text{RAS}}$  pin (*row-address strobe*—the bar indicates that the signal is active when it is low). This tells the DRAM to send an entire row of data (thousands of bits) into the DRAM's sense amplifiers (circuits that detect and amplify the tiny logic signals represented by the electric charges in the row's storage cells). This typically takes a few tens of nanoseconds, and the step may have already been done (the row or page could already be open or activated, meaning that the sense amps might already have valid data in them).

Once the sense amps have recovered the values, and the bitlines are pulled to the appropriate logic levels, the memory controller performs the last step, which is to *read* the column (*column* being the name given to the data subset of the row that is desired), by setting the chip-select signal to activate the set of DRAMs comprising the desired bank,<sup>2</sup> sending the column address and bank identifier over the address bus, and signaling the DRAM's  $\overline{\text{CAS}}$  pin (*column-address strobe*—like  $\overline{\text{RAS}}$ , the bar indicates that it is

<sup>2</sup>This step is necessary for SDRAMs; it is not performed for older, asynchronous DRAMs (it is subsumed by the earlier chip-select accompanying the RAS).

active when low). This causes only a few select bits<sup>3</sup> in the sense amplifiers to be connected to the output drivers, where they will be driven onto the data bus. Reading the column data takes on the order of tens of nanoseconds. When the memory controller receives the data, it forwards the data to the microprocessor.

The process of transmitting the address in two different steps (i.e., separately transmitted row and column addresses) is unlike that of SRAMs. Initially, DRAMs had minimal I/O pin counts because the manufacturing cost was dominated by the number of I/O pins in the package. This desire to limit I/O pins has had a long-term effect on DRAM architecture; the address pins for most DRAMs are still multiplexed, meaning that two different portions of a data address are sent over the same pins at different times, as opposed to using more address pins and sending the entire address at once.

Most computer systems have a special signal that acts much like a heartbeat and is called the *clock*. A clock transmits a continuous signal with regular intervals of “high” and “low” values. It is usually illustrated as a square wave or semi-square wave with each period identical to the next, as shown in Figure 7.9. The upward portion of the square wave is called the *positive* or *rising edge* of the clock, and the downward portion of the square wave is called the *negative* or *falling edge* of the clock. The primary clock in a computer system is called the system clock or global clock, and it typically resides on the motherboard (the PCB that contains the microprocessor and memory bus). The system clock drives the microprocessor and memory controller and many of the associated peripheral devices directly. If the clock drives the DRAMs directly, the DRAMs are called *synchronous DRAMs*. If the clock does not drive the DRAMs directly, the DRAMs are called *asynchronous DRAMs*. In a synchronous DRAM, steps internal to the DRAM happen in time with one or more edges of this clock. In an asynchronous DRAM, operative steps internal to the DRAM happen when the memory controller commands the DRAM to act, and those commands typically happen in time with one or more edges of the system clock.

## 7.2 Evolution of the DRAM Architecture

In the 1980s and 1990s, the conventional DRAM interface started to become a performance bottleneck in high-performance as well as desktop systems. The improvement in the speed and performance of microprocessors was significantly outpacing the improvement in speed and performance of DRAM chips. As a consequence, the DRAM interface began to evolve, and a number of revolutionary proposals [Przybylski 1996] were made as well. In most cases, what was considered evolutionary or revolutionary was the proposed *interface* (the mechanism by which the microprocessor accesses the DRAM). The DRAM core (i.e., what is pictured in Figure 7.3) remains essentially unchanged.

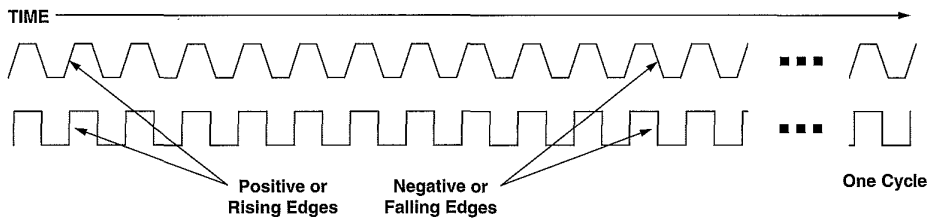
Figure 7.10 shows the evolution of the basic DRAM architecture from *clocked* to the conventional *asynchronous* to *fast page mode* (FPM) to *extended data-out* (EDO) to *burst-mode EDO* (BEDO) to *synchronous* (SDRAM). The figure shows each as a stylized DRAM in terms of the memory array, the sense amplifiers, and the column multiplexer (as well as additional components if appropriate).

As far as the first evolutionary path is concerned (asynchronous through SDRAM), the changes have largely been structural in nature, have been relatively minor in terms of cost and physical implementation, and have targeted increased throughput. Since SDRAM, there has been a profusion of designs proffered by the DRAM industry, and we lump these new DRAMs into two categories: those targeting reduced latency and those targeting increased throughput.

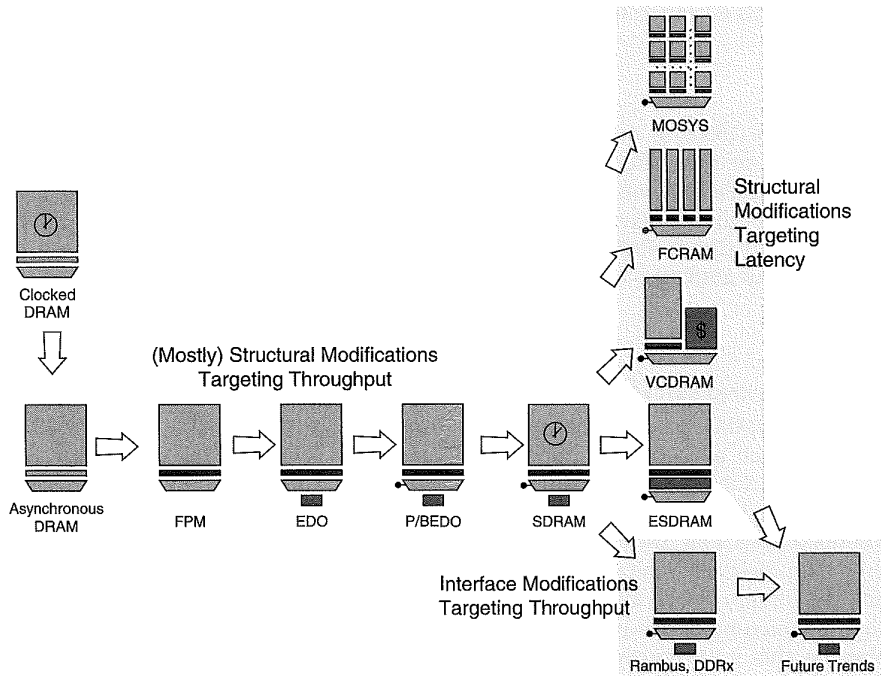
### 7.2.1 Structural Modifications Targeting Throughput

Compared to the conventional DRAM, FPM simply allows the row to remain open across multiple CAS commands, requiring very little additional circuitry. To this, EDO changes the output drivers to become output latches so that they hold the data valid on the

<sup>3</sup>One bit in a “x1” DRAM, two bits in a “x2” DRAM, four bits in a “x4” DRAM, etc.



**FIGURE 7.9:** Example clock signals. Clocks are typically shown as square waves (bottom) or sort of square waves (top). They repeat *ad infinitum*, and the repeating shape is called a clock cycle. The two clocks pictured above have the same frequency—the number of cycles in a given time period.



**FIGURE 7.10:** Evolution of the DRAM architecture. To the original DRAM design, composed of an array, a block of sense amps, and a column multiplexor, the fast page mode (FPM) design adds the ability to hold the contents of the sense amps valid over multiple column accesses. To the FPM design, extended data-out (EDO) design adds an output latch after the column multiplexor. To the EDO design, the burst EDO (BEDO) design adds a counter that optionally drives the column-select address latch. To the BEDO design, the synchronous DRAM (SDRAM) design adds a clock signal that drives both row-select and column-select circuitry (not just the column-select address latch).

bus for a longer period of time. To this, BEDO adds an internal counter that drives the address latch so that the memory controller does not need to supply a new address to the DRAM on every  $\overline{\text{CAS}}$  command if the desired address is simply one off from the previous  $\overline{\text{CAS}}$  command. Thus, in BEDO, the DRAM's column-select circuitry is driven from an internally generated signal, not an externally generated signal; the source of the control signal is close to the circuitry that it controls in space and therefore time, and this makes the timing of the circuit's activation more precise. Finally, SDRAM takes this perspective one step further and drives all internal circuitry (row select, column select, data read-out) by a clock, as opposed to the  $\overline{\text{RAS}}$  and  $\overline{\text{CAS}}$  strobes. The following paragraphs describe this evolution in more detail.

### Clocked DRAM

The earliest DRAMs (1960s to mid-1970s, before de facto standardization) were often clocked [Rhoden 2002, Sussman 2002, Padgett and Newman 1974]; DRAM commands were driven by a periodic clock signal. Figure 7.10 shows a stylized DRAM in terms of the memory array, the sense amplifiers, and the column multiplexer.

### The Conventional Asynchronous DRAM

In the mid-1970s, DRAMs moved to the asynchronous design with which most people are familiar. These DRAMs, like the clocked versions before them, require that every single access go through all of the steps described previously: for every access, the bitlines need to be precharged, the row needs to be activated, and the column is read out after row activation. Even if the microprocessor wants to request the same data row that it previously requested, the entire process (row activation followed by column read/write) must be repeated. Once the column is read, the row is deactivated or closed, and the bitlines are precharged. For the next request, the entire process is repeated, even if the same datum is requested twice in succession. By convention and

circuit design, both  $\overline{\text{RAS}}$  and  $\overline{\text{CAS}}$  must rise in unison. For example, one cannot hold  $\overline{\text{RAS}}$  low while toggling  $\overline{\text{CAS}}$ . Figure 7.11 illustrates the timing for the conventional asynchronous DRAM.

### Fast Page Mode DRAM (FPM DRAM)

FPM DRAM implements page mode, an improvement on conventional DRAM in which the row address is held constant and data from multiple columns is read from the sense amplifiers. This simply lifts the restriction described in the previous paragraph: the memory controller may hold  $\overline{\text{RAS}}$  low while toggling  $\overline{\text{CAS}}$ , thereby creating a de facto cache out of the data held active in the sense amplifiers. The data held in the sense amps form an “open page” that can be accessed relatively quickly. This speeds up successive accesses to the same row of the DRAM core, as is very common in computer systems (the term is locality of reference and indicates that oftentimes memory requests that are nearby in time are also nearby in the memory-address space and would therefore likely lie within the same DRAM row). Figure 7.12 gives the timing for FPM reads.

### Extended Data-Out DRAM (EDO DRAM)

EDO DRAM, sometimes referred to as hyper-page mode DRAM, adds a few transistors to the output drivers of an FPM DRAM to create a latch between the sense amps and the output pins of the DRAM. This latch holds the output pin state and permits the  $\overline{\text{CAS}}$  to rapidly deassert, allowing the memory array to begin precharging sooner. In addition, the latch in the output path also implies that the data on the outputs of the DRAM circuit remain valid longer into the next clock phase, relative to previous DRAM architectures (thus the name “extended data-out”). By permitting the memory array to begin precharging sooner, the addition of a latch allows EDO DRAM to operate faster than FPM DRAM. EDO enables the microprocessor to access memory at least 10 to 15% faster than with FPM [Kingston 2000, Cuppu et al. 1999, 2001]. Figure 7.13 gives the timing for an EDO read.

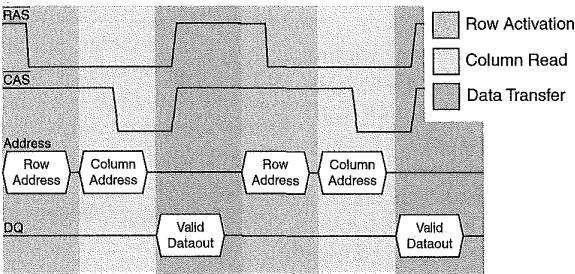


FIGURE 7.11: Read timing for the asynchronous DRAM.

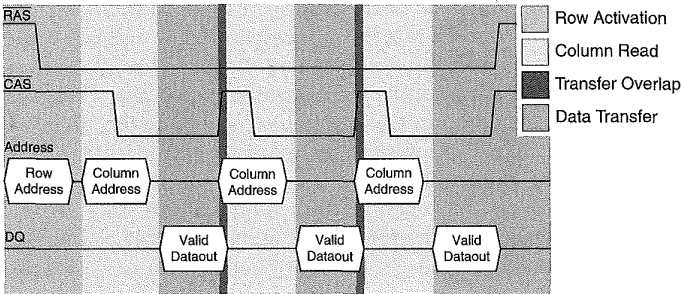


FIGURE 7.12: FPM read timing. The FPM allows the DRAM controller to hold a row constant and receive multiple columns in rapid succession.

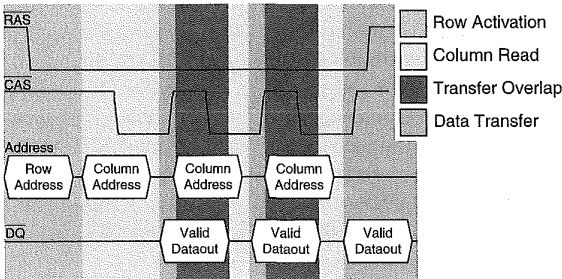


FIGURE 7.13: EDO read timing. The output latch in EDO DRAM allows more overlap between column access and data transfer than in FPM.

### Burst-Mode EDO DRAM (BEDO DRAM)

Although BEDO DRAM never reached the volume of production that EDO and SDRAM did, it was positioned to be the next-generation DRAM after EDO [Micron 1995]. BEDO builds on EDO DRAM by adding the concept of “bursting” contiguous blocks of data from an activated row each time a new column address is sent to the DRAM chip. An internal counter was added that first accepts the incoming address and then increments that value on every successive toggling of  $\overline{\text{CAS}}$ , driving the incremented value into the column-address latch. With each toggle of the  $\overline{\text{CAS}}$ , the DRAM chip sends the next sequential column of data onto the bus. In previous DRAMs, the column-address latch was driven by an externally generated address signal. By eliminating the need to send successive column addresses over the bus to drive a burst of data in response to each microprocessor request, BEDO eliminates a significant amount of timing uncertainty between successive addresses, thereby increasing the rate at which data can be read from the DRAM. In practice, the minimum cycle time for driving the output bus was reduced by roughly 30% compared to EDO DRAM [Prince 2000], thereby increasing bandwidth proportionally. Figure 7.14 gives the timing for a BEDO read.

### IBM’s High-Speed Toggle Mode DRAM

IBM’s High-Speed Toggle Mode (“toggle mode”) is a high-speed DRAM interface designed and fabricated in the late 1980s and presented at the International Solid-State Circuits Conference in February 1990 [Kalter et al. 1990a]. In September 1990, IBM presented toggle mode to JEDEC as an option for the next-generation DRAM architecture (minutes of JC-42.3 meeting 55). Toggle mode transmits data to and from a DRAM on both edges of a high-speed data strobe rather than transferring data on a single edge of the strobe. The strobe was very high speed for its day; Kalter reports a 10-ns data cycle time—an effective 100 MHz data rate—in 1990 [Kalter

et al. 1990b]. The term “toggle” is probably derived from its implementation: to obtain twice the normal data rate,<sup>4</sup> one would toggle a signal pin which would cause the DRAM to toggle back and forth between two different (interleaved) output buffers, each of which would be pumping data out at half the speed of the strobe [Kalter et al. 1990b]. As proposed to JEDEC, it offered burst lengths of 4 or 8 bits of data per memory access.

### Synchronous DRAM (SDRAM)

Conventional, FPM, and EDO DRAM are controlled asynchronously by the memory controller. Therefore, in theory, the memory latency and data toggle rate can be some fractional number of microprocessor clock cycles.<sup>5</sup> More importantly, what makes the DRAM asynchronous is that the memory controller’s RAS and  $\overline{\text{CAS}}$  signals directly control latches internal to the DRAM, and those signals can arrive at the DRAM’s pins at any time. An alternative is to make the DRAM interface synchronous such that requests can only arrive at regular intervals. This allows the latches internal to the DRAM to be controlled by an internal clock signal. The primary benefit is similar to that seen in BEDO: by associating all data and control transfer with a clock signal, the timing of events is made more predictable. Such a scheme, by definition, has less skew. Reduction in skew means that the system can potentially achieve faster turnaround on requests, thereby yielding higher throughput. A timing diagram for synchronous DRAM is shown in Figure 7.15. Like BEDO DRAMs, SDRAMs support the concept of a burst mode; SDRAM devices have a programmable register that holds a burst length. The DRAM uses this to determine how many columns to output over successive cycles; SDRAM may therefore return many bytes over several cycles per request. One advantage of this is the elimination of the timing signals (i.e., toggling  $\overline{\text{CAS}}$ ) for each successive burst, which

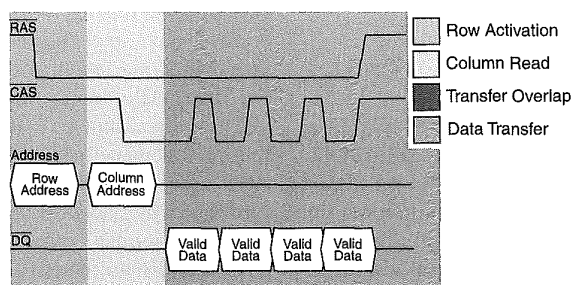
<sup>4</sup>The term “normal” implies the data cycling at half the data-strobe rate.

<sup>5</sup>In practice, this is not the case, as the memory controller and DRAM subsystem are driven by the system clock, which typically has a period that is an integral multiple of the microprocessor’s clock.

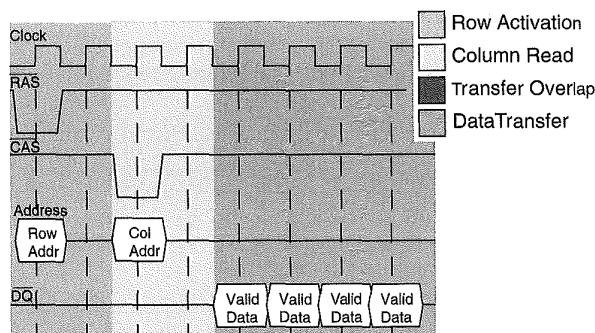
reduces the command bandwidth used. The underlying architecture of the SDRAM core is the same as in a conventional DRAM.

The evolutionary changes made to the DRAM interface up to and including BEDO have been relatively inexpensive, especially when considering the pay-off: FPM was essentially free compared to the conventional design, EDO simply added a latch, and BEDO added a counter and mux. Each of these

evolutionary changes added only a small amount of logic, yet each improved upon its predecessor by as much as 30% in terms of system performance [Cuppu et al. 1999, 2001]. Though SDRAM represented a more significant cost in implementation and offered no performance improvement over BEDO at the same clock speeds,<sup>6</sup> the presence of a source-synchronous data strobe in its interface (in this case, the global clock signal) would allow SDRAM to scale to much



**FIGURE 7.14:** BEDO read timing. By driving the column-address latch from an internal counter rather than an external signal, the minimum cycle time for driving the output bus was reduced by roughly 30% over EDO.



**FIGURE 7.15:** SDR SDRAM read operation clock diagram (CAS-2).

<sup>6</sup>By some estimates, BEDO actually had higher performance than SDRAM [Williams 2001].

higher switching speeds more easily than the earlier asynchronous DRAM interfaces such as FPM and EDO.<sup>7</sup> Note that this benefit applies to any interface with a source-synchronous data strobe signal, whether the interface is synchronous or asynchronous, and therefore, an asynchronous burst-mode DRAM with source-synchronous data strobe could have scaled to higher switching speeds just as easily—witness the February 1990 presentation of a working 100-MHz asynchronous burst-mode part from IBM, which used a dedicated pin to transfer the source-synchronous data strobe [Kalter 1990a, b].

### 7.2.2 Interface Modifications Targeting Throughput

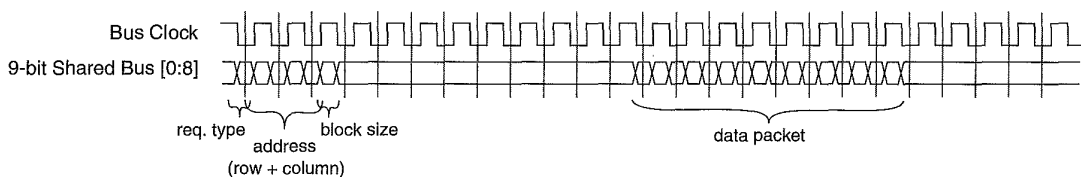
Since the appearance of SDRAM in the mid-1990s, there has been a large profusion of novel DRAM architectures proposed in an apparent attempt by DRAM manufacturers to make DRAM less of a commodity [Dipert 2000]. One reason for the profusion of competing designs is that we have apparently run out of the same sort of “free” ideas that drove the earlier DRAM evolution. Since BEDO, there has been no architecture proposed that provides a 30% performance advantage at near-zero cost; all proposals have been relatively expensive. As Dipert suggests, there is no clear heads-above-the-rest winner yet because many schemes seem to lie along a linear relationship between additional cost of implementation and realized performance gain. Over time, the market will most likely decide the winner;

those DRAM proposals that provide sub-linear performance gains relative to their implementation cost will be relegated to zero or near-zero market share.

#### Rambus DRAM (RDRAM, Concurrent RDRAM, and Direct RDRAM)

Rambus DRAM (RDRAM) is very different from traditional main memory. It uses a bus that is significantly narrower than the traditional bus, and, at least in its initial incarnation, it does not use dedicated address, control, data, and chip-select portions of the bus. Instead, the bus is fully multiplexed: the address, control, data, and chip-select information all travel over the same set of electrical wires but at different times. The bus is 1 byte wide, runs at 250 Mhz, and transfers data on both clock edges to achieve a theoretical peak bandwidth of 500 MB/s. Transactions occur on the bus using a split request/response protocol. The packet transactions resemble network request/response pairs: first an address/control packet is driven, which contains the entire address (row address and column address), and then the data is driven. Different transactions can require different numbers of cycles, depending on the transaction type, location of the data within the device, number of devices on the channel, etc. Figure 7.16 shows a typical read transaction with an arbitrary latency.

Because of the bus's design—being a single bus and not composed of separate segments dedicated to separate functions—only one transaction can use



**FIGURE 7.16:** Rambus read clock diagram for block size 16. The original Rambus design (from the 1990 patent application) had but a single bus multiplexed between data and address/control. The request packet is six “cycles,” where a cycle is one beat of a cycle, not one full period of a cycle.

<sup>7</sup>Making an interface synchronous is a well-known technique to simplify the interface and to ease scaling to higher switching speeds [Stone 1982].

the bus during any given cycle. This limits the bus's potential *concurrency* (its ability to do multiple things simultaneously). Due to this limitation, the original RDRAM design was not considered well suited to the PC main memory market [Przybylski 1996], and the interface was redesigned in the mid-1990s to support more concurrency.

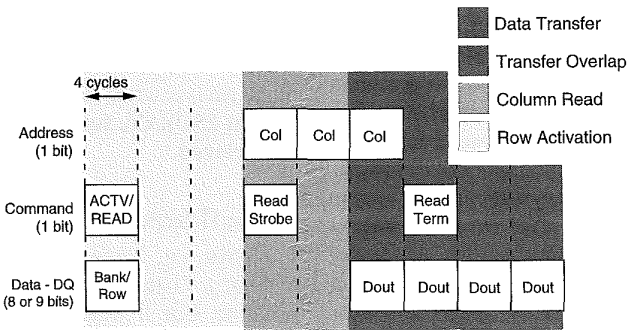
Specifically, with the introduction of “Concurrent RDRAM,” the bus was divided into separate address, command, and data segments reminiscent of a JEDEC-style DRAM organization. The data segment of the bus remained 1 byte wide, and to this was added a 1-bit address segment and a 1-bit control segment. By having three separate, dedicated segments of the bus, one could perform potentially three separate, simultaneous actions on the bus. This divided and dedicated arrangement simplified transaction scheduling and increased performance over RDRAM accordingly. Note that at this point, Rambus also moved to a four clock cycle period, referred to as an octcycle. Figure 7.17 gives a timing diagram for a read transaction.

One of the few limitations to the “Concurrent” design was that the data bus sometimes carried a brief packet of address information, because the 1-bit address bit was too narrow. This limitation has been removed in Rambus’ latest DRAMs. The divided arrangement introduced in Concurrent RDRAM has been carried over into the most recent incarnation of

RDRAM, called “Direct RDRAM,” which increases the width of the data segment to 2 bytes, the width of the address segment to 5 bits, and the width of the control segment to 3 bits. These segments remain separate and dedicated—similar to a JEDEC-style organization—and the control and address segments are wide enough that the data segment of the bus never needs to carry anything but data, thereby increasing data throughput on the channel. Bus operating speeds have also changed over the years, and the latest designs are more than double the original speeds (500 MHz bus frequency). Each half-row buffer in Direct RDRAM is shared between adjacent banks, which implies that adjacent banks cannot be active simultaneously. This organization has the result of increasing the row buffer miss rate as compared to having one open row per bank, but it reduces the cost by reducing the die area occupied by the row buffers, compared to 16 full row buffers. Figure 7.18 gives a timing diagram for a read operation.

Double Data Rate DRAM (DDR SDRAM)

Double data rate (DDR) SDRAM is the modern equivalent of IBM’s High-Speed Toggle Mode. DDR doubles the data bandwidth available from single data rate SDRAM by transferring data at both edges of the clock (i.e., both the rising edge and the falling



**FIGURE 7.17:** Concurrent RDRAM read operation. Concurrent RDRAMs transfer on both edges of a fast clock and use a 1-byte data bus multiplexed between data and addresses.

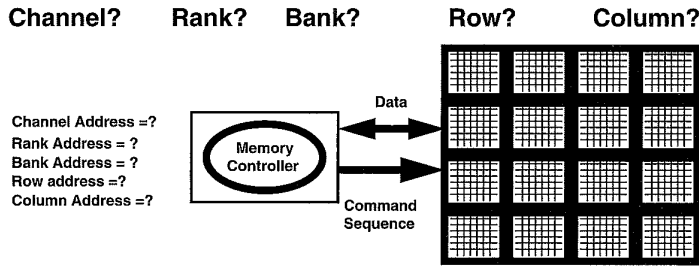


FIGURE 10.1: Multiple DRAM devices connected to a processor through a DRAM memory controller.

memory system in terms of system storage capacity, operating data rates, access latency, and sustainable bandwidth characteristics. It is therefore of great importance that the organization of multiple DRAM devices into larger memory systems be examined in detail. However, the absence of commonly accepted nomenclature has hindered the examination of DRAM memory-system organizations. Without a common basis of well-defined nomenclature, technical articles and data sheets sometimes succeed in introducing confusion rather than clarity into discussions on DRAM memory systems. In one example, a technical data sheet for a system controller used the word *bank* in two bulleted items on the same page to mean two different things. In this data sheet, one bulleted item proclaimed that the system controller could support 6 *banks* (of DRAM devices). Then, several bulleted items later, the same data sheet stated that the same system controller could support SDRAM devices with 4 *banks*. In a second example, an article in a well-respected technical journal examined the then-new i875P system controller from Intel and proceeded to discuss the performance advantage of the system controller due to the fact that the i875P system controller could control 2 *banks* of DRAM devices (it can control two entire channels).

In these two examples, the word *bank* was used to mean three different things. While the meaning

of the word *bank* can be inferred from the context in each case, the overloading and repeated use of the word introduces unnecessary confusion into discussions about DRAM memory systems. In this section, the usage of channel, rank, bank, row, and column is defined, and discussions in this and subsequent chapters will conform to the usage in this chapter.

10.2.1 Channel

Figure 10.2 shows three different system controllers with slightly different configurations of the DRAM memory system. In Figure 10.2, each system controller has a single *DRAM memory controller* (DMC), and each DRAM memory controller controls a single channel of memory. In the example labelled as the *typical system controller*, the system controller controls a single 64-bit-wide channel. In modern DRAM memory systems, commodity DRAM memory modules are standardized with 64-bit-wide data busses, and the 64-bit data bus width of the memory module matches the data bus width of the typical personal computer system controller.<sup>1</sup> In the example labelled as *Intel i875P system controller*, the system controller connects to a single channel of DRAM with a 128-bit-wide data bus. However, since commodity DRAM modules have 64-bit-wide data busses,

<sup>1</sup>Commodity memory modules designed for error correcting memory systems are standardized with a 72-bit-wide data bus.

the i875P system controller requires matching pairs of 64-bit wide memory modules to operate with the 128-bit-wide data bus. The paired-memory module configuration of the i875P is often referred to as a *dual channel* configuration. However, since there is only one memory controller, and since both memory modules operate in lockstep to store and retrieve data through the 128-bit-wide data bus, the paired-memory module configuration is, logically, a 128-bit-wide single channel memory system. Also, similar to SDRAM and DDR SDRAM memory systems, standard

Direct RDRAM memory modules are designed with 16-bit-wide data busses, and high-performance system controllers that use Direct RDRAM, such as the Intel i850 system controller, use matched pairs of Direct RDRAM memory modules to form a 32-bit-wide channel that operates in lockstep across the two physical channels of memory.

In contrast to system controllers that use a single DRAM memory controller to control the entire memory system, Figure 10.3 shows that the Alpha EV7 processor and the Intel i925x system controller each have

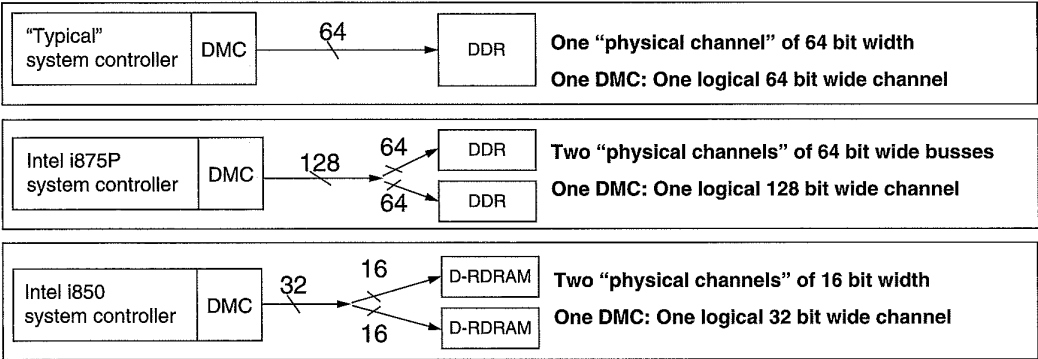


FIGURE 10.2: Systems with a single memory controller and different data bus widths.

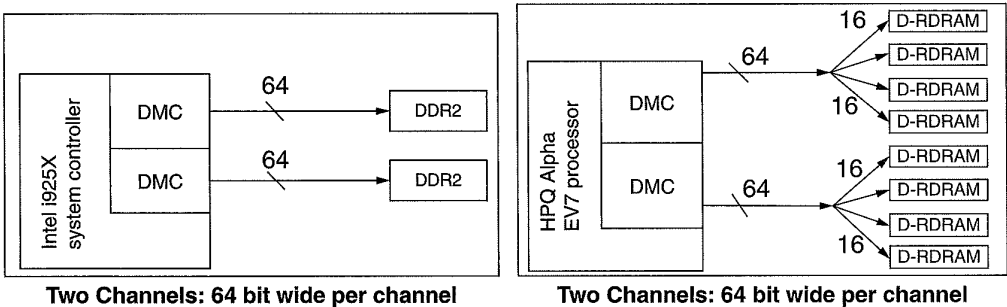


FIGURE 10.3: Systems with two independent memory controllers and two logical channels of memory.

two DRAM controllers that independently control 64-bit-wide data busses.<sup>2</sup> The use of independent DRAM memory controllers can lead to higher sustainable bandwidth characteristics, since the narrower channels lead to longer data bursts per cacheline request, and the various inefficiencies dictated by DRAM-access protocols can be better amortized. As a result, newer system controllers are often designed with multiple memory controllers despite the additional die cost.

Modern memory systems with one DRAM memory controller and multiple physical channels of DRAM devices such as those illustrated in Figure 10.2 are typically designed with the physical channels operating in lockstep with respect to each other. However, there are two variations to the single-controller-multiple-physical-channel configuration. One variation of the single-controller-multiple-physical-channel configuration is that some system controllers, such as the Intel i875P system controller, allow the use of mismatched pairs of memory modules in the different physical channels. In such a case, the i875P system controller operates

in an *asymmetric* mode and independently controls the physical channels of DRAM modules. However, since there is only one DRAM memory controller, the multiple physical channels of mismatched memory modules cannot be accessed concurrently, and only one channel of memory can be accessed at any given instance in time. In the asymmetric configuration, the maximum system bandwidth is the maximum bandwidth of a single physical channel.

A second variation of the single-controller-multiple-physical-channel configuration can be found in high-performance FPM DRAM memory systems that were designed prior to the emergence of SDRAM-type DRAM devices that can burst out multiple columns of data with a given column access command. Figure 10.4 illustrates a sample timing diagram of a column access in an SDRAM memory system. Figure 10.4 shows that an SDRAM device is able to return a burst of multiple columns of data for a single column access command. However, an FPM DRAM device supported neither single-access-multiple-burst capability nor the ability to pipeline multiple column access commands. As a result, FPM DRAM

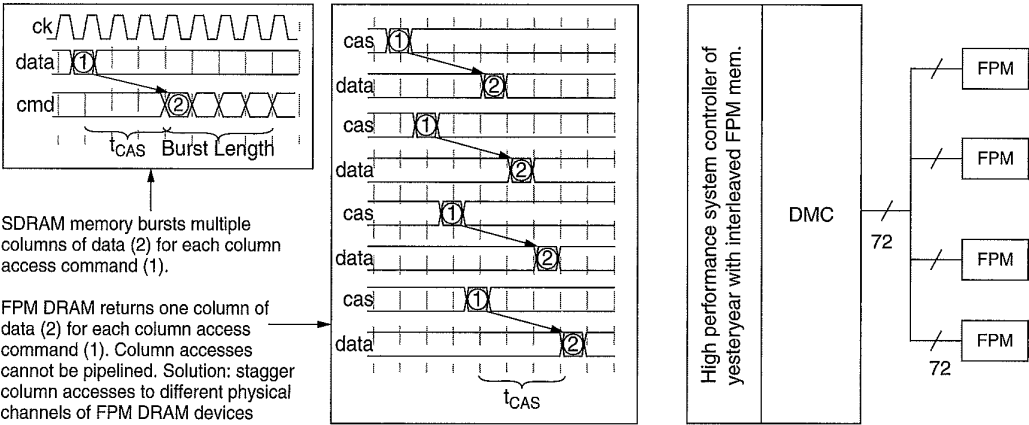


FIGURE 10.4: High-performance memory controllers with four channels of interleaved FPM DRAM devices.

<sup>2</sup>Ignoring additional bitwidths used for error correction and cache directory.

devices need multiple column accesses to retrieve the multiple columns of data for a given cacheline access; column accesses that cannot be pipelined to a single FPM DRAM device.

One solution deployed to overcome the shortcomings of FPM DRAM devices is the use of multiple FPM DRAM channels operating in an interleaved fashion. Figure 10.4 also shows how a sophisticated FPM DRAM controller can send multiple column accesses to different physical channels of memory so that the data for the respective column accesses appears on the data bus in consecutive cycles. In this configuration, the multiple FPM DRAM channels can provide the sustained throughput required in high-performance workstations and servers before the appearance of modern synchronous DRAM devices that can burst through multiple columns of data in consecutive cycles.

### 10.2.2 Rank

Figure 10.5 shows a memory system populated with 2 ranks of DRAM devices. Essentially, a *rank* of memory is a “bank” of one or more DRAM devices that operate in lockstep in response to a given command. However, the word *bank* has already been used to describe the number of independent DRAM arrays within a DRAM device. To lessen the confusion associated with overloading the nomenclature, the word *rank* is now used to denote a set of DRAM devices that operate in lockstep to respond to a given command in a memory system.

Figure 10.5 illustrates a configuration of 2 ranks of DRAM devices in a classical DRAM memory system topology. In the classical DRAM memory system topology, address and command busses are connected to every DRAM device in the memory system,

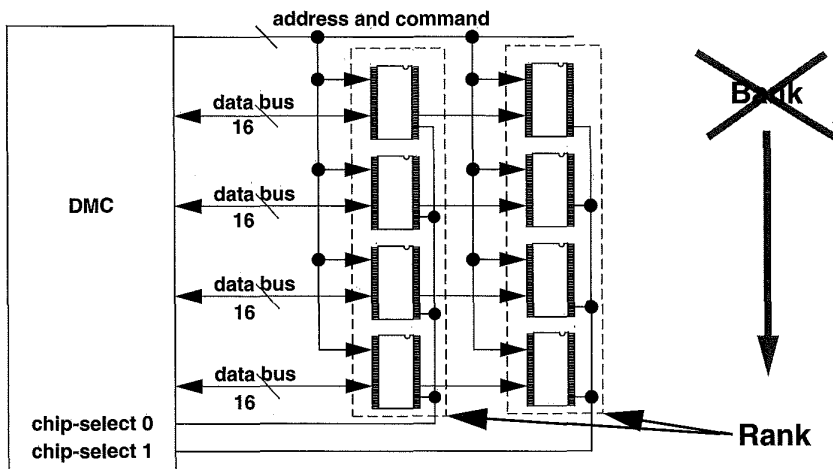
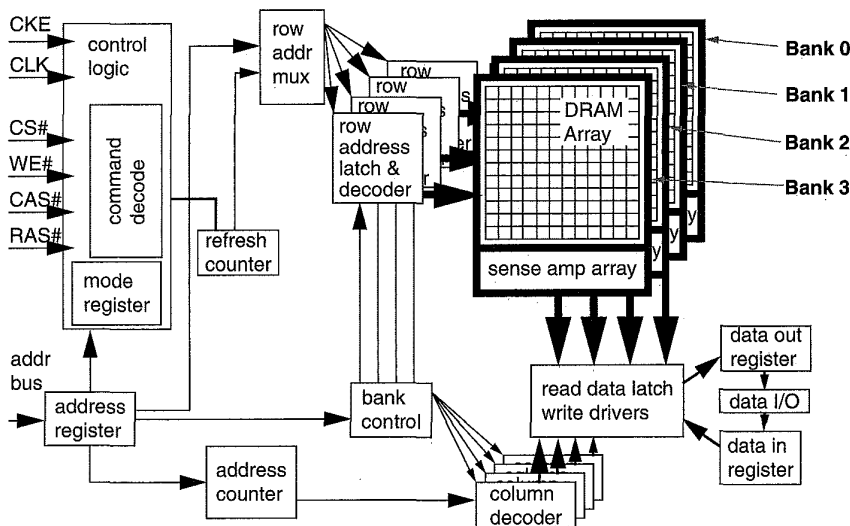


FIGURE 10.5: Memory system with 2 ranks of DRAM devices.



but the wide data bus is partitioned and connected to different DRAM devices. The memory controller in this classical system topology then uses chip-select signals to select the appropriate rank of DRAM devices to respond to a given command.

### 10.2.3 Bank

and different physical channels of memory. In this chapter, the word *bank* is only used to denote a set of independent memory arrays inside a DRAM device.

time. Multiple banks in a given DRAM device can also be precharged or refreshed in parallel, depending on the design of the DRAM device.

#### 10.2.4 Row

In DRAM devices, a *row* is simply a group of storage cells that are activated in parallel in response to a row activation command. In DRAM memory systems that utilize the conventional system topology such as SDRAM, DDR SDRAM, and DDR2 SDRAM memory systems, multiple DRAM devices are typically connected in parallel in a given rank of memory. Figure 10.7 shows how DRAM devices can be connected in parallel to form a rank of memory. The effect of DRAM devices connected as ranks of DRAM devices that operate in lockstep is that a row activation command will activate the same addressed row in all DRAM devices in a given rank of memory. This arrangement means that the size of a row—from the perspective of the memory controller—is simply the size of a row in a given DRAM device multiplied by

the number of DRAM devices in a given rank, and a DRAM row spans across the multiple DRAM devices of a given rank of memory.

A *row* is also referred to as a DRAM page, since a row activation command in essence caches a page of memory at the sense amplifiers until a subsequent precharge command is issued by the DRAM memory controller. Various schemes have been proposed to take advantage of locality at the DRAM page level. However, one problem with the exploitation of locality at the DRAM page level is that the size of the DRAM page depends on the configuration of the DRAM device and memory modules, rather than the architectural page size of the processor.

#### 10.2.5 Column

In DRAM memory systems, a column of data is the smallest addressable unit of memory. Figure 10.8 illustrates that, in memory systems such as SDRAM and DDRx<sup>3</sup> SDRAM with topology similar to the memory system illustrated in Figure 10.5, the size of

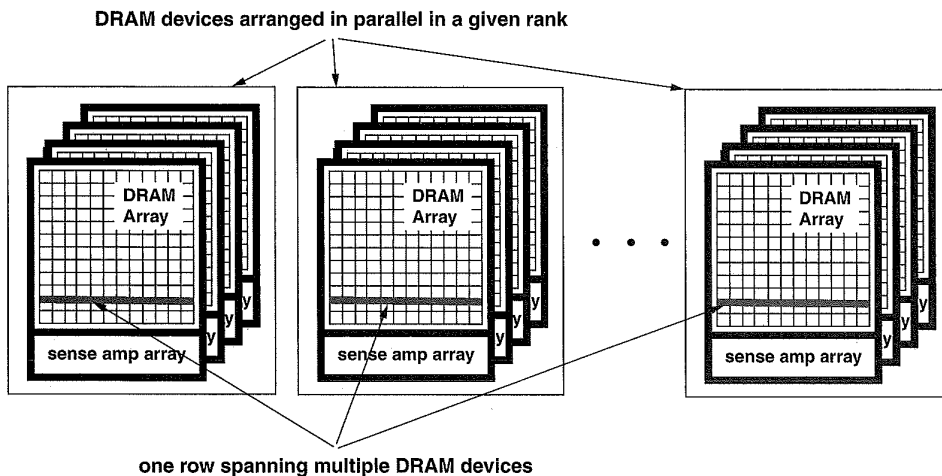
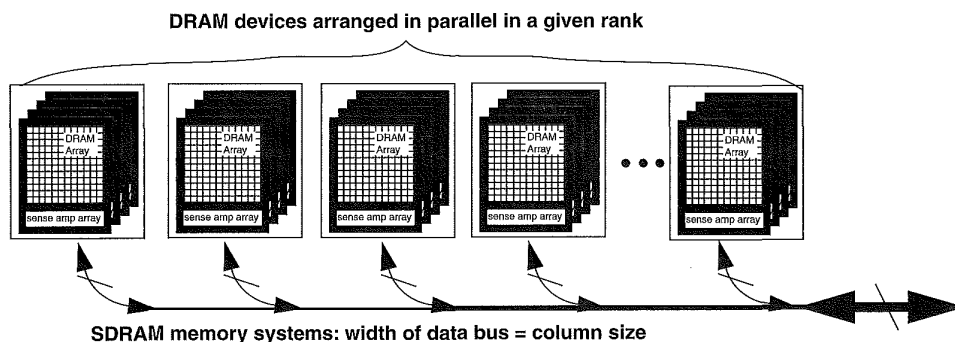


FIGURE 10.7: Generic DRAM devices with 4 banks, 8196 rows, 512 columns per row, and 16 data bits per column.

<sup>3</sup>DDR<sub>x</sub> denotes DDR SDRAM and evolutionary DDR memory systems such as DDR2 and DDR3 SDRAM memory systems, inclusively.



**FIGURE 10.8:** Classical DRAM system topology; width of data bus equals column size.

a column of data is the same as the width of the data bus. In a Direct RDRAM device, a column is defined as 16 bytes of data, and each read command fetches a single column of data 16 bytes in length from each physical channel of Direct RDRAM devices.

A *beat* is simply a data transition on the data bus. In SDRAM memory systems, there is one data transition per clock cycle, so one beat of data is transferred per clock cycle. In DDRx SDRAM memory systems, two data transfers can occur in each clock cycle, so two beats of data are transferred in a single clock cycle. The use of the beat terminology avoids overloading the word *cycle* in DDRx SDRAM devices.

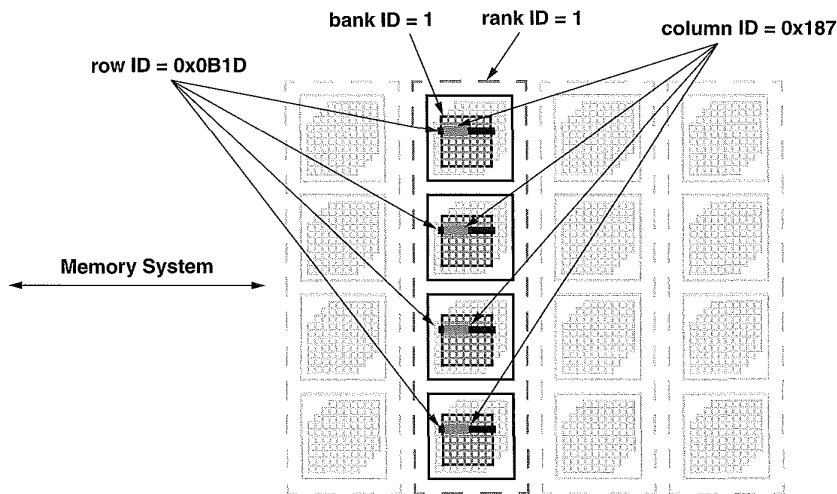
In DDRx SDRAM memory systems, each column access command fetches multiple columns of data depending on the programmed burst length. For example, in a DDR2 DRAM device, each memory read command returns a minimum of 4 columns of data. The distinction between a DDR2 device returning a minimum burst length of 4 *beats* of data and a Direct RDRAM device returning a single column of data over 8 beats is that the DDR2 device accepts the address of a specific column and returns the requested columns in different orders depending on the programmed behavior of the DRAM device. In this manner, each column is separately addressable. In contrast, Direct RDRAM devices do not reorder data within a given burst, and a 16-byte burst from a single channel of

Direct RDRAM devices is transmitted in order and treated as a single column of data.

### 10.2.6 Memory System Organization: An Example

Figure 10.9 illustrates a DRAM memory system with 4 ranks of memory, where each rank of memory consists of 4 devices connected in parallel, each device contains 4 banks of DRAM arrays internally, each bank contains 8192 rows, and each row consists of 512 columns of data. To access data in a DRAM-based memory system, the DRAM memory controller accepts a physical address and breaks down the address into respective address fields that point to the specific channel, rank, bank, row, and column where the data is located.

Although Figure 10.9 illustrates a uniformly organized memory system, memory system organizations of many computer systems, particularly end-user configurable systems, may be typically non-uniformly organized. The reason that the DRAM memory systems organizations in many computer systems are typically non-uniform is because most computer systems are designed to allow end-users to upgrade the capacity of the memory system by inserting and removing commodity memory modules. To support memory capacity upgrades by the end-user, DRAM controllers have to be designed to



**FIGURE 10.9:** Location of data in a DRAM memory system.

flexibly adapt to different configurations of DRAM devices and modules that the end-user could place into the computer system. This support is provided for through the use of address range registers whose functionality is examined separately in the chapter on memory controllers.

### 10.3 Memory Modules

The first generations of computer systems allowed end-users to increase memory capacity by providing sockets on the system board where additional DRAM devices could be inserted. The use of sockets on the system board made sense in the era where the price of DRAM devices was quite expensive relative to the cost of the sockets on the system board. In these early computer systems, system boards were typically designed with sockets that allowed end-users to remove and insert individual DRAM devices, usually contained in dual in-line packages (DIPs). The process of memory upgrade was cumbersome and

difficult, as DRAM devices had to be individually removed and inserted into each socket. Pins on the DRAM devices may have been bent and not visually detected as such. Defective DRAM chips were difficult to locate, and routing of sockets for a large memory system required large surface areas on the system board. Moreover, it was physically possible to place DRAM devices in the wrong orientation in the socket—180° from the intended placement. Correct placement with proper orientation depended on clearly labelled sockets, clearly labelled devices, and an end-user that paid careful attention while inserting the devices into the sockets.<sup>4</sup> The solution to the problems associated with memory upgradability was the creation and use of memory modules.

Memory modules are essentially miniature system boards that hold a number of DRAM devices. Memory modules provide an abstraction at the module interface so that different manufacturers can manufacture memory upgrades for a given computer system with different DRAM devices. DRAM memory

<sup>4</sup>The author of this text can personally attest to the consequences of inserting chips into sockets with incorrect orientation.

modules also reduce the complexity of the memory upgrade process. Instead of the removal and insertion of individual DRAM chips, memory upgrades with modules containing multiple DRAM chips can be quickly and easily inserted into and removed from a module socket. The first generations of memory modules typically consisted of specially created, system-specific memory modules that a given computer manufacturer used in a given computer system. Over the years, memory modules have obtained a level of sophistication, and they are now specified as a part of the memory-system definition process.

### 10.3.1 Single In-line Memory Module (SIMM)

In the late 1980s and early 1990s, the personal computer industry first standardized on the use of 30-pin SIMMs and then later moved to 72-pin SIMMs. SIMMs, or *Single In-line Memory Modules*, are referred to as such due to the fact that the contacts on either side of the bottom of the module are electrically identical.

A 30-pin SIMM provides interconnects to 8 or 9 signals on the data bus, as well as power, ground, address, command, and chip-select signal lines between the system board and the DRAM devices. A 72-pin SIMM provides interconnects to 32 to 36 signals on the data bus in addition to the power, ground, address, command, and chip-select signal lines. Typically, DRAM devices on a 30 pin, 1 Megabyte SIMM collectively provide a 9-bit, parity protected data bus interface to the memory system. Personal computer systems in the late 1980s typically used sets of four matching 30-pin SIMMs to provide a 36-bit-wide memory interface to support parity checking by the memory controller. Then, as the personal computer system moved to support memory systems with wider data busses, the 30-pin SIMM was replaced by 72-pin SIMMs in the early 1990s.

### 10.3.2 Dual In-line Memory Module (DIMM)

In the late 1990s, as the personal computer industry transitioned from FPM/EDO DRAM to SDRAM, 72-pin SIMMs were, in turn, phased out in favor of *Dual In-line Memory Modules* (DIMMs). DIMMs are physically larger than SIMMs and provide a

64- or 72-bit-wide data bus interface to the memory system. The difference between a SIMM and a DIMM is that contacts on either side of a DIMM are electrically different. The electrically different contacts allow a denser routing of electrical signals from the system board through the connector interface to the memory module.

Typically, a DIMM designed for the commodity desktop market contains little more than the DRAM devices and passive resistor and capacitors. These DIMMs are not buffered on either the address path from the memory controller to the DRAM devices or the datapath between the DRAM devices and the memory controller. Consequently, these DIMMs are also referred to as *Unbuffered DIMMs* (UDIMMs).

### 10.3.3 Registered Memory Module (RDIMM)

To meet the widely varying requirements of systems with end-user configurable memory systems, memory modules of varying capacity and timing characteristics are needed in addition to the typical UDIMM. For example, workstations and servers typically require larger memory capacity than those seen for the desktop computer systems. The problem associated with large memory capacity memory modules is that the large number of DRAM devices in a memory system tends to overload the various multi-drop busses. The large number of DRAM devices, in turn, creates the loading problem on the various address, command, and data busses.

*Registered Dual In-line Memory Modules* (RDIMMs) alleviate the issue of electrical loading of large numbers of DRAM devices in a large memory system through the use of registers that buffer the address and control signals at the interface of the memory module. Figure 10.10 illustrates that registered memory modules use registers at the interface of the memory module to buffer the address and control signals. In this manner, the registers greatly reduce the number of electrical loads that a memory controller must drive directly, and the signal interconnects in the memory system are divided into two separate segments: between the memory controller and the register and between the register and DRAM devices. The segmentation allows timing characteristics of the memory system to be optimized by limiting

the number of electrical loads, as well as by reducing the path lengths of the critical control signals in individual segments of the memory system. However, the drawback to the use of the registered latches on a memory module is that the buffering of the address and control signals introduces delays into the memory-access latency, and the cost of ensuring signal

integrity in a large memory system is paid in terms of additional latency for all memory transactions.

10.3.4 Small Outline DIMM (SO-DIMM)

Over the years, memory module design has become ever more sophisticated with each new generation of DRAM devices. Currently, different module specifications exist as standardized, multi-source components that an end-user can purchase and reasonably expect trouble-free compatibility between memory modules manufactured by different module manufacturers at different times. To ensure system-level compatibility, memory modules are specified as part of the memory system standards definition process. More specifically, different types of memory modules are specified, with each targeting different markets. Typically, UDIMMs are used in desktop computers, RDIMMs are used in workstation and server systems, and the *Small Outline Dual In-line Memory Module* (SO-DIMM) has been designed to fit into the limited space found in mobile notebook computers.

Figure 10.11 shows the standardized placement of eight DDR2 SDRAM devices in *Fine Ball Grid Array* (FBGA) packages along with the required serial termination resistors and decoupling capacitors on

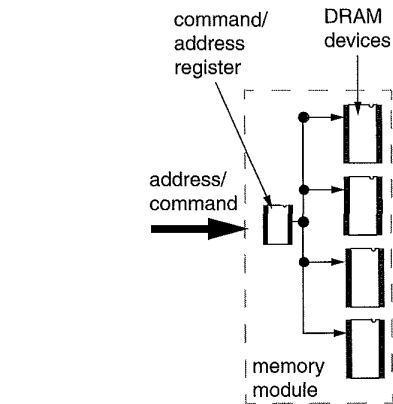


FIGURE 10.10: Registered latches buffer the address and command and also introduce additional latency into the DRAM access.

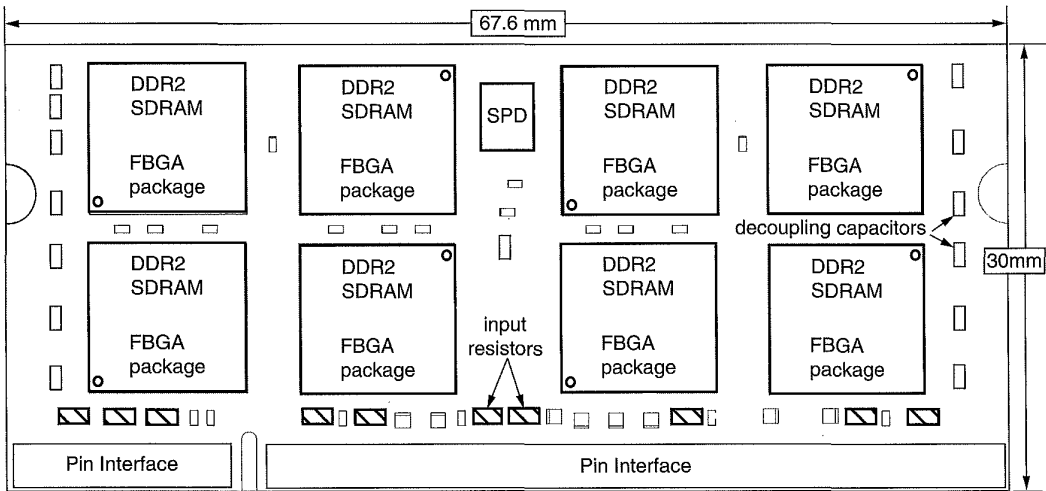


FIGURE 10.11: Component placement specification for a DDR2 SO-DIMM.

a 200-pin SO-DIMM. Figure 10.11 shows that the outline of the SO-DIMM is standardized with specific dimensions: 30 mm × 67.6 mm. The specification of the SO-DIMM dimension illustrates the point that as part of the effort to ensure system-level compatibility between different memory modules and system boards, mechanical and electrical characteristics of SO-DIMMs, UDIMMs, and RDIMMs have been carefully defined. Currently, commodity DRAM devices and memory modules are defined through long and arduous standards-setting processes by DRAM device manufacturers and computer-system design houses.

The standards-setting process enables DRAM manufacturers to produce DRAM devices that are functionally compatible. The standards-setting process further enables memory-module manufacturers to take the functionally compatible DRAM devices and construct memory modules that are functionally compatible with each other. Ultimately, the multi-level standardization enables end-users to freely purchase memory modules from different module manufacturers, using DRAM devices from different DRAM manufacturers, and to enjoy reasonably trouble-free interoperability. Currently, standard commodity DRAM devices and memory modules are specified through the industry organization known as the JEDEC Solid-State Technology Association.<sup>5</sup>

Finally, to further minimize problems in achieving trouble-free compatibility between different DRAM devices and memory module manufacturers, JEDEC provides reference designs to memory module manufacturers, complete with memory module raw card specification, signal trace routings, and a bill of materials. The reference designs further enable memory module manufacturers to minimize their expenditure of engineering resources in the process to create and validate memory module designs, thus lowering the barrier of entry to the manufacturing of high-quality memory modules and enhancing competition in the memory module manufacturing business.

### 10.3.5 Memory Module Organization

Modern DRAM memory systems often support large varieties of memory modules to give end-users the flexibility of selecting and configuring the desired memory capacity. Since the price of DRAM devices fluctuates depending on the unpredictable commodity market, one memory module organization may be less expensive to manufacture than another organization at a given instance in time, while the reverse may be true at a different instance in time. As a result, a memory system that supports different configurations of memory modules allows end-users the flexibility to purchase and use the most economically organized memory module. However, one issue that memory-system design engineers must account for in providing the flexibility of memory system configuration to the end-user is that the flexibility translates into large combinations of memory modules that may be placed into the memory system at one time. Moreover, multiple organizations often exist for a given memory module capacity, and memory system design engineers must often account for not only different combinations of memory modules of different capacities, but also different modules of different organizations for a given capacity.

Table 10.1 shows that a 128-MB memory module can be constructed from a combination of 16 64-Mbit DRAM devices, 8 128-Mbit DRAM devices, or 4 256-Mbit DRAM devices. Table 10.1 shows that the different memory-module organizations not only use different numbers of DRAM devices, but also present different numbers of rows and columns to the memory controller. To access the memory on the memory module, the DRAM controller must recognize and support the organization of the memory module inserted by the end-user into the memory system. In some cases, new generations of DRAM devices can enable memory module organizations that a memory controller was not designed to support, and incompatibility follows naturally.

<sup>5</sup>JEDEC was once known as the Joint Electron Device Engineering Council.

10.3.6 Serial Presence Detect (SPD)

Memory modules have gradually evolved as each generation of new memory modules gains additional levels of sophistication and complexity. Table 10.1 shows that a DRAM memory module can be organized as multiple ranks of DRAM devices on the same memory module, with each rank consisting of multiple DRAM devices, and the memory module can have differing numbers of rows and columns. What is not shown in Table 10.1 is that each DRAM memory module may, in fact, have different minimum timing characteristics in terms of minimum  $t_{CAS}$ ,  $t_{RAS}$ ,  $t_{RCD}$ , and  $t_{RP}$  latencies. The variability of the DRAM modules, in turn, increases the complexity that a memory-system design engineer must deal with.

To reduce the complexity and eliminate the confusion involved in the memory upgrading process, the solution adopted by the computer industry is to store the configuration information of the memory module on a read-only memory device whose content can be retrieved by the memory controller as part of the system initialization process. In this manner, the memory controller can obtain the configuration and timing parameters required to optimally access data from DRAM devices on the memory module. Figure 10.12 shows the image of a small flash memory device on a DIMM. The small read-only memory device is known as a *Serial Presence Detect* (SPD) device, and it stores a wide range of variations that can exist between different memory modules. Table 10.2 shows some parameters and values that are stored in the SPD of the DDR SDRAM memory module.

TABLE 10.1 Four different configurations for a 128-MB SDRAM memory module

Capacity	Device Density	Number of Ranks	Devices per Rank	Device Width	Number of Banks	Number of Rows	Number of Columns
128 MB	64 Mbit	1	16	x4	4	4096	1024
128 MB	64 Mbit	2	8	x8	4	4096	512
128 MB	128 Mbit	1	8	x8	4	4096	1024
128 MB	256 Mbit	1	4	x16	4	8192	512

serial presence detect (SPD)

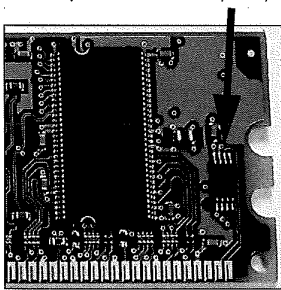


FIGURE 10.12: The SPD device stores memory module configuration information.

TABLE 10.2 Sample parameter values stored in SPD

Configuration	Value (interpreted)
DRAM type	DDR SDRAM
No. of row addresses	16384
No. of column addresses	1024
No. of banks	4
Data rate	400
Module type	ECC
CAS latency	3

10.4 Memory System Topology

In Figure 10.13, a memory system where 16 DRAM devices are connected to a single DRAM controller is shown. In Figure 10.13, the 16 DRAM devices are organized into 4 separate *ranks* of memory. Although all 16 DRAM devices are connected to the same DRAM controller, different numbers of DRAM devices are connected to different networks for the unidirectional address and command bus, the bidirectional data bus, and the unidirectional chip-select lines. In this topology, when a command is issued, electrical signals on the address and command busses are sent to all 16 DRAM devices in the memory system, but the separate chip-select signal selects a set of 4 DRAM devices in a single rank to provide the data for a read command or receive the data for a write command. In this topology, each DRAM device in a given rank of memory is also connected to a subset of the width of the data bus along with three other DRAM devices in different ranks of memory.

Memory system topology determines the signal path lengths and electrical loading characteristics in the memory system. As a result, designers of modern high-performance DRAM memory systems must pay close attention to the topology and organizations of the DRAM memory system. However, due to the evolutionary nature of the memory system, the classic system topology described above has remained

essentially unchanged for Fast Page Mode DRAM (FPM), Synchronous DRAM (SDRAM), and Dual Data Rate SDRAM (DDR) memory systems. Furthermore, variants of the classical topology with fewer ranks are expected to be used for DDR2 and DDR3 memory systems.

10.4.1 Direct RDRAM System Topology

One memory system with a topology dramatically different from the classical topology is the Direct RDRAM memory system. In Figure 10.14, four Direct RDRAM devices are shown connected to a single Direct RDRAM memory controller. Figure 10.14 shows that in a Direct RDRAM memory system, the DRAM devices are connected to a well-matched network of interconnects where the clocking network, the data bus, and the command busses are all path-length matched by design. The benefit of the well-matched interconnection network is that signal skew is minimal by design, and electrical signaling rates in the Direct RDRAM memory system can be increased to higher frequencies than a memory system with the classic memory system topology. Modern DRAM systems with conventional multi-rank topology can also match the raw signaling rates of a Direct RDRAM memory system. However, the drawback is that idle cycles must be designed into the access protocol and devoted to system-level synchronization. As a result,

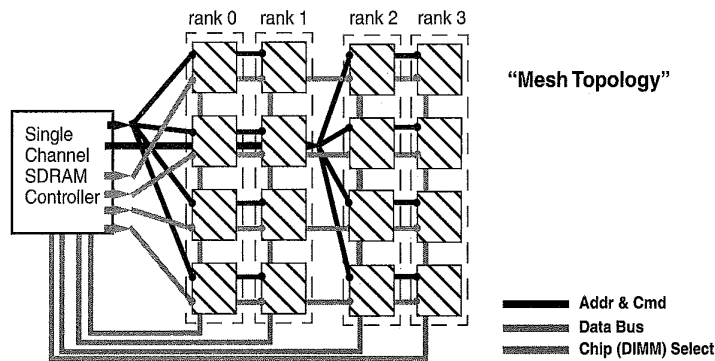


FIGURE 10.13: Topology of a generic DRAM memory system.

# MEMORY SYSTEMS

## Cache, DRAM, Disk

**Bruce Jacob**, University of Maryland at College Park  
**Spencer W. Ng**, Hitachi Global Storage Technologies  
**David T. Wang**, MetaRAM

*"Memory Systems: Cache, DRAM, Disk fills a huge void in the literature about modern computer architecture...Jacob, Ng, and Wang have created one of the truly great technology books that turns reading about bits and bytes into an exciting journey towards understanding technology."*

— Michael Schuette, VP of Technology Development at OCZ Technology

*"Memory Systems: Cache, DRAM, Disk is remarkable in both its scope and depth.... This book will doubtless be the definitive reference for students and designers of memory systems for many years to come."*

— Jim Smith, University of Wisconsin – Madison

The performance gap between processor and memory speeds has increased dramatically. System designers have talked for over a decade about "hitting the memory wall," a condition in which overall system speed is significantly compromised by the memory system's inability to keep pace with the processor. That condition is today's reality—the memory system has become the most critical performance bottleneck, and memory subsystems design is now the greatest challenge facing design engineers.

*Memory Systems: Cache, DRAM, Disk* is the first book to cover comprehensively the logical design and operation, physical design and operation, performance characteristics and resulting design trade-offs, and the energy consumption of modern memory hierarchies. This book describes the physical design and detailed software emulation of the entire memory hierarchy, from cache to disk and prepares designers to tackle the challenging optimization problems that result from the side effects that can appear at any point in the entire memory hierarchy.

### CONTENT HIGHLIGHTS

- Provides a comprehensive, in-depth understanding of all levels of the system hierarchy—cache, DRAM, and disk.
- Describes a holistic approach for evaluating the system-level effects of all design choices.
- Includes a link to the open-source software DRAMsim, a detailed and accurate parameter-driven simulator of modern DRAM-based memory systems that models performance and energy consumption for each component.

### ABOUT THE AUTHORS

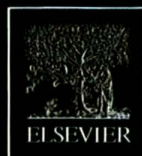
**Bruce Jacob** is an Associate Professor of Electrical and Computer Engineering at the University of Maryland at College Park.

**Spencer W. Ng** is a senior technical staff member with Hitachi Global Storage Technologies. He has been a researcher in the field of storage for over twenty years, and is the holder of about twenty issued U.S. patents.

**David T. Wang** is a senior technical staff member at MetaRAM, a memory systems startup in Silicon Valley.

Online support materials for this book are available at  
[textbooks.elsevier.com/9780123797513](http://textbooks.elsevier.com/9780123797513)

Cover photograph: istockphotos.com/ Mypokcik



MORGAN KAUFMANN PUBLISHERS  
AN IMPRINT OF ELSEVIER SCIENCE  
[www.mkp.com](http://www.mkp.com)

